Catching the head, tail, and everything in between: a streaming algorithm for the degree distribution

Olivia Simpson*
osimpson@ucsd.edu
University of California, San Diego

C. Seshadhri
scomandu@ucsc.edu
University of California, Santa Cruz

Andrew McGregor
mcgregor@cs.umass.edu
University of Massachusetts, Amherst

Abstract

The degree distribution is one of the most fundamental graph properties of interest for real-world graphs. It has been widely observed in numerous domains that graphs typically have a *tailed* or *scale-free* degree distribution. While the average degree is usually quite small, the variance is quite high and there are vertices with degrees at all scales. We focus on the problem of approximating the degree distribution of a large streaming graph, with small storage. We design an algorithm `headtail`, whose main novelty is a new estimator of infrequent degrees using truncated geometric random variables. We give a mathematical analysis of `headtail` and show that it has excellent behavior in practice. We can process streams with millions of edges with storage less than 1% and get extremely accurate approximations for *all* scales in the degree distribution.

We also introduce a new notion of *Relative Hausdorff* distance between tailed histograms. Existing notions of distances between distributions are not suitable, since they ignore infrequent degrees in the tail. The Relative Hausdorff distance measures deviations at all scales, and is a more suitable distance for comparing degree distributions. By tracking this new measure, we are able to give strong empirical evidence of the convergence of `headtail`.

1 Introduction

Graphs are a natural abstraction for any data set with entities and relationship between them. Popular examples include online social networks such as Facebook and Twitter; transportation networks; biological networks such as protein-protein interaction and metabolic networks; and communication networks such as the internet and telephone and email networks. Many of these graphs are most naturally represented by a *stream of edges*. Especially for social and communication networks, each edge has an associated timestamp, and the graph is basically an aggregate of all these edges over some time window. Such streams are typically quite massive; social networks like Facebook and Twitter can generate billions of communication links in a day [1, 2]. A publicly available HTTP request dataset has billions of requests [3]. The scale of these data sizes has

---

*Work was done while the author was an intern at Sandia National Laboratories, Livermore.

led to interest in *small-space streaming algorithms*. Such algorithms accurately compute specific properties of the total graph, using a memory footprint that is orders of magnitude smaller in size.

Arguably, one of the most important properties of real-world networks is the *degree distribution*. Seminal papers in massive graph analysis studied precisely this quantity [4, 5, 6]. The study of degree distributions is probably the birthplace of real-world network analysis. It has been found to be relevant for graph modeling, network resilience, and algorithmics [7, 8, 9, 10, 11, 12, 13]. One of the key discoveries of network analysis is the presence of *scale-free* or heavy-tailed degree distributions. The average degree of a node is usually small, but there are nodes with degrees at all scales. The very notion of a *scale-free network* has entered the common parlance because of its relevance to network analysis [14].

1.1 Problem statement

The input is a stream of edges $e_1, e_2, \ldots, e_m$ without any repetitions. The graph created by these edges is denoted $G = (V, E)$. For convenience, we set $V = [n]$, though the labels may be from some arbitrary discrete universe. We do not assume that the algorithm knows n and m, the number of vertices and edges respectively. Each edge is represented by a pair (u, v) of vertex labels.

For vertex $v \in V$, d_v denotes its degree (the number of neighbors of v). We set $n(d)$ to be the number of vertices of degree d, and $N(d)$ to be the number of vertices of degree at least d. In math, $N(d) = \sum_{r \geq d} n_r$. It is convenient for us to work with unnormalized raw counts, so we deal with histograms rather than distributions. We denote the sequence $\{n(d)\}$ by the *degree histogram* (dh) and $\{N(d)\}$ is the *complementary cumulative degree histogram*[1] (ccdh). When $\{n(d)\}$ is normalized by n, it is called the degree distribution. We focus on the ccdh, instead of the dh. Typically, the dh is quite noisy in real data, and the ccdh has the added benefit of being monotonically decreasing. (Focus on the ccdh is standard for fitting procedures [15].)

We study the problem of approximating the ccdh of G using a *small-space one-pass streaming algorithm*. Such an algorithm has some limited memory, denoted M. It sees the edges in stream order, and on seeing edge e_t, updates the memory M. The algorithm cannot access older edges, and M is typically order of magnitudes smaller than the size of the stream. At the end of the stream, the algorithm reports a sequence $\{\widehat{N}(d)\}$, an approximation to the ccdh of G.

We make no assumption on the ordering of edges. We do not consider edge deletions or edge repetitions. (This is the standard model used in most work on practical streaming algorithms.)

1.2 Challenges

How does a small-space algorithm estimate the degree distribution at all scales? The degree distribution involves degrees at "all" scales: many low degree vertices, some intermediate degree vertices, and few very high degree vertices. Look at Fig. 1a for the ccdh of a router topology network. The average degree is 20, but there are vertices with degrees up to $50,000$. The count of low degree vertices is easy to estimate, since a simple random sample of vertices gives a good estimate. Intermediate and high degrees pose a problem. There are few such vertices but it is critical to sample their count accurately. There is a huge literature on estimating distribution properties of a stream of items: frequent items, distribution moments, distinct items, etc. [17, 18, 19]. (We discuss in depth later.) But these only give specific properties of the distribution. None of these

---

[1]This is often called the cumulative degree distribution, but that is counter to the standard definition for probability distributions.

(a) as-Skitter: $n = 1.7M$, $m = 11M$, storage $= 31K$

(b) com-LiveJournal: $n = 4M$, $m = 34M$, storage $= 200K$
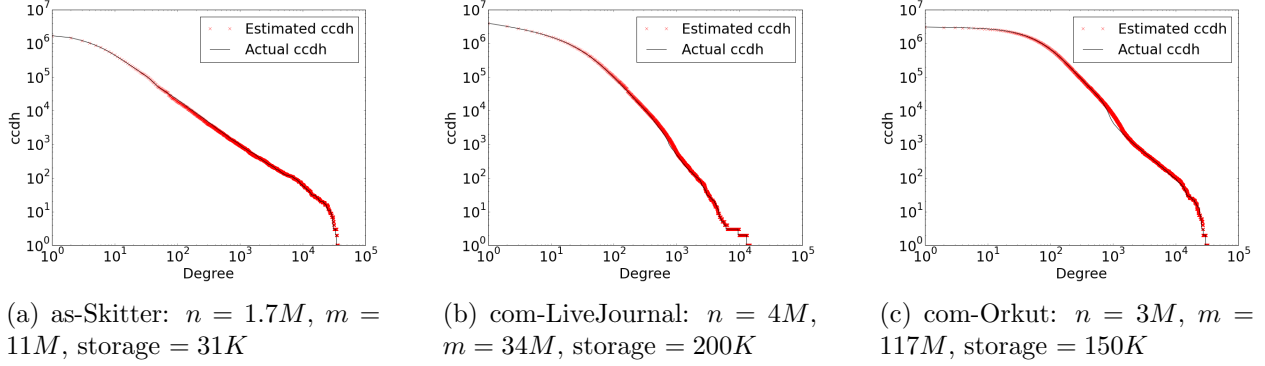
(c) com-Orkut: $n = 3M$, $m = 117M$, storage $= 150K$

Figure 1: The output ccdh of `headtail` on three different input graphs from the SNAP [16] collection. In each case, the storage is less 1% of the stream (and less than 5% of the number of vertices). Observe the near identical match with the true ccdh.

methods can get frequency estimates at *all scales*, ranging continuously from (frequent) low degrees to (infrequent) high degrees.

How to quantitatively compare (cumulative) degree distributions? How do we actually assert that our algorithm is any good? One can use standard statistical distance measures like Kolmogorov-Smirnov. Yet these measures typically ignore the tail since it contains a negligible fraction of vertices. Consider the following examples. We take a clique of n vertices and a clique of $n - 1$ vertices. It is natural to say that their degree distributions are quite close, but no popular existing measure would assert that. On the other end, consider a star with n edges, and a matching with n edges. The degree distribution only differs at one "point", the vertex of degree n. Yet we would consider the degree distributions to be fundamentally different. Most statistical measures would say they are similar, since they differ at only a single outlier.

An intuitive notion of similarity is closeness in log-log plots, but how do we quantify such a concept? One might try to approximate degree distributions by closed-form, but fitting procedures are notoriously tricky for tailed distributions and subject to much error [15].

1.3 Main results

The algorithm `headtail`: Our main contribution is a new small-space algorithm `headtail` that estimates the ccdh of an input graph stream. The novelty is a new estimator for infrequent degree counts, which is combined with standard sampling to give ccdh estimates at all scales. We represent the sampling of `headtail` through certain *truncated geometric* random variables. An analysis of their behavior provides the right "correction" factors to infer the ccdh from our sampling. We provide a detailed mathematical analysis of `headtail` explaining why it accurately estimates the ccdh. Our analysis falls short of a complete proof, and we rely on some heuristic arguments for the full argument.

Relative Hausdorff distance: We introduce a new notion of distance between ccdhs (technically, between any two histograms) called the *Relative Hausdorff (RH) distance*. This distance avoids the pitfalls of standard measures, and is able to capture the closeness at *all scales*. Intuitively, a small RH-distance implies that *every* point in one ccdh is "close" (up to relative error) to some point in the other ccdh. Put another way, both ccdhs agree at all scales, and agree on

3

outliers. While this condition is quite stringent, RH distance is flexible enough to allow for minor errors. It gives a concrete way of quantifying the quality of `headtail`, and empirically establishing convergence of our estimate.

Empirical behavior of `headtail`: We run `headtail` on a wide variety of public graph datasets. It gives excellent estimates of the ccdh in all our tests, for storage less than 1% of the stream. We show example outputs in Fig. 1, for three different input graphs. In each case, observe the near perfect match with the true ccdh, at *all* degrees. We compute the RH distance for numerous runs and demonstrate convergence of `headtail`'s output with increasing storage. In all our runs, storage around 1% of the stream is sufficient for excellent match in ccdhs (and also for low RH-distance).

1.4 Related Work

Note that we can frame our problem in terms of general histogram estimation. If one views the input as a stream of vertex labels, then the dh (and ccdh) is the histogram of label frequencies. There is much work on understanding frequencies in a discrete stream, but as we detail below, none of this work solves the problem of estimating the ccdh.

Finding frequent items, aka "heavy hitters," is a classic problem in the data stream model. Cormode and Hadjieleftheriou [19] compare three of the most important algorithms: the *frequent* algorithm [20, 21, 22], the *lossy counting* algorithm [23], and the *space saving* algorithm [24].[2] For large degrees, these approaches will give accurate results, but the error term dwarfs the degree at smaller scales. We demonstrate this empirically in Section 5. Much work has been done in approximating frequency moments [27, 17, 18, 19], but they do not give an estimate for multiple scales. Nor has this work been implemented in practice for large data sets.

Rather than just finding frequent items, Korn et al. [28] attempt to estimate the entire distribution of elements in the stream. However, in contrast to our work, their approach assumes that the distribution comes from a parameterized family of distributions, e.g., the distribution is Zipfian, and then focuses on estimating the relevant parameters. This approach is only applicable for graphs where the degree distribution is already relatively well understood. Despite much study and claims, there are *no* conclusive closed-form formulae for real-world degree distributions. The classic power law fitting work of Clauset et al. [15] argues why most previous methods are not statistically robust, and how one needs strong independence assumptions to get rigorous results. Therefore, `headtail` makes no closed form assumption on the input stream.

Over the last ten years, there has been a growing body of work focused on processing graphs in the data stream model. See [29] for a summary of recent work on graph streaming and sketching. This work has included problems such as the number of triangles and related quantities such as the transitivity coefficient [30, 31, 32], estimating the connectivity properties of a graph [33], and solving combinatorial problems such as computing large matchings [34, 35]. Cormode and Muthukrishnan considered estimating properties of the degree distribution in multigraphs but not the distribution itself[36].

Closest to this work is the series of graph sampling papers by Ahmed et al. [37, 38, 39, 32]. Their work focuses on estimating many properties (as opposed to a single property) with a fixed sampling method, and they study various sampling schemes. The results on estimating ccdhs typically use

---

[2]Other popular algorithms such as CountSketch [25] and CountMin [26] enable frequent items to be identified when the frequency of an item may be incremented and decremented.

20-30% of the stream, with weaker empirical results [37]. The recent Graph Sample and Hold framework gives extremely strong results for triangle counting [32], but is not applied for the ccdh. This technique is closely related to an approach for estimating frequency moments [27, 40]. Our sampling approach is also similar, and our main contribution is in the actual estimation procedure.

2 The algorithm

The algorithm `headtail` has two parts: `update` and `estimate`. The procedure `update` is called for every edge in the stream, and simply updates the data structures. The procedure `estimate` is called at the end of the stream to get an estimate of $\{N(d)\}$. In what follows, the subscript h refers to "head" and t is "tail".

The algorithm `headtail` requires two parameters, p_h and p_t, which are probabilities. These decide the storage requirements of the algorithm, as explained later. For convenience, we will assume these are global variables, and will not pass them around to each function.

We will assume the existence of a hash function $hash$ that maps strings uniformly to $[0, 1]$.

Data Structures: There are two sets of vertices S_h and S_t, and corresponding maps $\text{ct}_h : S_h \mapsto \mathbb{N}$ and $\text{ct}_t : S_t \mapsto \mathbb{N}$. Again, we assume these are global variables.

The procedure `update`: This updates the data structures for each edge in the stream. Consider edge (u, v) in the stream. If $v \in S_h$, the $\text{ct}_h(v)$ is incremented (analogously for S_t). Now for the critical difference between S_h and S_t. If $v \notin S_h$ and if $\text{hash}(v) \leq p_h$, then v is added to S_h. If $v \notin S_t$: we insert v to S_t with probability p_t. (The entire operation above is also done for u.) Note the difference: for S_h, we essentially flip a random coin for the vertex. For S_t, we flip a coin for the edge. Intuitively, S_h is maintaining a uniform random set of vertices. On the other hand, S_t maintains sample of vertices biased towards higher degree.

The procedure `estimate`: This procedure uses $S_h, S_t, \text{ct}_h, \text{ct}_t$ to output an estimate $\{\widehat{N}(d)\}$ for the ccdh of G. We set $\text{C}_h(r)$ to be the number of vertices in S_h with $\text{ct}_h(\cdot)$ value of r (similarly for $\text{C}_t(r)$). One can think of this as the "observed" degree distribution. The scaling of $\text{C}_h(r)$ is straightforward: we simply consider $\text{C}_h(r)/p_h$ to be an estimate of $n(r)$. By summing these appropriately, we get an estimate (the head estimate) of $N(r)$.

For C_t, we first do an additive "correction". So we set $\widetilde{\text{C}}_t(r) = \text{C}_t(r - \ell(r))$, where $\ell(r)$ is a correction factor. The explanation of this factor is provided in Section 3. Then, we do a biased scaling and consider $\widetilde{\text{C}}_t(r)/(1 - (1 - p_t)^r)$ as an estimate of $n(r)$. Again, by taking partial sums, we have an estimate (the tail estimate) of $N(r)$.

Observe that we have two different estimates of $N(r)$. We prove in our mathematical analysis that the former is accurate for the head of the distribution, while the latter is appropriate for the tail. This distinction is made by d_{thr}, which is chosen to ensure that the first estimate has low variance. Hence, for all degrees less than d_{thr}, we use the head estimate, and for the remaining, we use the tail estimate.

We now give a formal description of the algorithm.

For fixed $p_t \in (0, 1)$, we define $\ell(r)$ to be:

$$\left\lceil \frac{1 - p_t - (1 - p_t)^{r+1} - rp_t(1 - p_t)^r}{p_t(1 - (1 - p_t)^r)} \right\rceil$$

Algorithm 1: headtail(p_h, p_t)

---

1 Initialize empty sets S_h and S_t and empty mappings ct_h and ct_t.

2 For each edge $e_i = (u, v)$ in the stream,

3   Call update(u, v).

4 Call estimate to get output estimate for $\{N(d)\}$.

Algorithm 2: update(u, v)

---

1 If $u \in S_h$, increment $\text{ct}_h(u)$.

2 If $u \notin S_h$: if $hash[u] < p_h$, insert u in S_h and set $\text{ct}_h(u) = 1$.

3 If $u \in S_t$, increment $\text{ct}_t(u)$.

4 If $u \notin S_t$: with probability p_t, insert u in S_t and set $\text{ct}_t(u) = 1$.

5 (Repeat above steps for v.)

Algorithm 3: estimate

---

1 Let $\text{C}_h(r)$ be the number of vertices in S_h with count exactly r. (Similarly, define $\text{C}_t(r)$).

2 For all counts r, set $\widetilde{\text{C}}_h(r) = \text{C}_h(r - \ell(r))$.

3 For all counts r:

4   Set $g_h(r) = \text{C}_h(r)/p_h$.

5   Set $g_t(r) = \widetilde{\text{C}}_t(r)/[1 - (1 - p_t)^r]$.

6 Set d_{thr} to be largest d such that $\sum_{r \geq d} g_h(r) \geq 50/p_h$.

7 For all degrees d:

8   If $d \leq d_{thr}$, set $\widehat{N}(d) = \sum_{r \geq d} g_h(r)$.

9   If $d > d_{thr}$, set $\widehat{N}(d) = \sum_{r \geq d} g_t(r)$.

---

3 Mathematical Analysis

We abstract out the behavior of the algorithm in a series of claims. We stress that all our theorems are independent of graph stream order, and hence `estimate` works for all orderings.

Definition 1. *For any positive integer s and $p \in (0,1)$, the* truncated geometric distribution *$TG_{p,s}$ has the pdf: $\forall 0 \leq k \leq s-1$, $\Pr[X = k] = p(1-p)^k / [1 - (1-p)^s]$.*

Observe that as $s \to \infty$, this is a standard geometric random variable.

Lemma 1. *For every $v \in [n]$, v is inserted in S_h independently with probability p_h. Conditioned on $v \in S_h$, $ct(v) = d_v$.*

Proof. We assume that hash is a uniform random function, so $hash(v)$ is uniformly distributed in $(0,1)$. The probability that $hash(v) \leq p_h$ is exactly p_h. Observe that if $hash(v) \leq p_h$, then v is inserted in S_h at the very first occurrence of v in the stream. Hence $ct(v) = d(v)$, whenever $v \in S_h$. $\square$

Lemma 2. *For every $v \in [n]$, v is inserted in S_t independently with probability $1 - (1-p_t)^{d_v}$. Conditioned on $v \in S_t$, $ct(v) = d_v - X$, where $X \sim TG_{p_t, d_v}$.*

Proof. There are d_v occurrences of v in the stream. The probability of v being added in the bth occurrence is $p_t(1-p_t)^{b-1}$. When this happens, $ct(v) = d_v - (b-1)$. The probability that v is never added is $\sum_{b=1}^{d_v} p_t(1-p_t)^{b-1} = (1-p_t)^{d_v}$. Conditioned on v being added to S_t, the probability of v being added in the bth occurrence is exactly $p_t(1-p_t)^{b-1} / [1 - (1-p_t)^{d_v}]$. So $b-1$ is distributed as TG_{p_t, d_v}. $\square$

Lemma 3. *The expected value of $X \sim TG_{p,d}$ is $\frac{1 - p - (1-p)^{d+1} - dp(1-p)^d}{p(1-(1-p)^d)}$.*

Proof. Using the bound for the sum of an arithmetico-geometric series:

$$\frac{p}{1-(1-p)^d} \sum_{k=0}^{d} k(1-p)^k = \frac{p}{1-(1-p)^d} \left(\frac{(1-d)(1-p)^d}{p} + \frac{(1-p) - (1-p)^d}{p^2} \right)$$

$$= \frac{1 - p - (1-p)^{d+1} - dp(1-p)^d}{p(1-(1-p)^d)}.$$

$\square$

This expression is exactly (up to rounding) $\ell(d)$. Conditioned on $v \in S_t$, $\mathbf{E}[ct(v)]$ is d_v minus a "loss" term, which is precisely the expression in Lemma 3. That should hopefully explain the use of $\ell(d)$ in our algorithm. We make the (admittedly wrong) assumption that *every* vertex of degree d in S_t "loses" exactly the expected loss. In other words, we assume that $ct(v)$ is $\mathbf{E}[ct(v)]$. To infer the number of degree d vertices in S_t, we add back the expected loss to each vertex in v. That is why we set $\widetilde{C}_t(r) = C_t(r - \ell(r))$.

It is fairly easy to bound the space and running time of `headtail`.

Theorem 4. *The expected space used by `headtail` is $O(p_h n + p_t m)$. The expected running time of `update` is $O(1)$, and the expected running time of `estimate` is $O(p_h n + p_t m)$.*

Proof. We will store all sets as hash tables, to ensure $O(1)$ updates. By Lemma 1, each vertex is added to S_h with probability p_h. Hence, the expected size of S_h is $O(p_h n)$. For each edge in the stream, we potentially add a vertex to S_t with probability p_t. Hence, the expected size of S_t is $O(p_t m)$. (This is a gross upper bound, and a refined bound based on Lemma 2 would be $\sum_d n(d)[1 - (1-p)^d]$.)

The processing of `update` only requires addition in set and count increments, and requires $O(1)$ time. The procedure `estimate` runs in time linear in the sets S_h and S_t. □

3.1 The estimators

For the analysis of our estimators, we need to introduce various error parameters. Natually, the actual implementation `estimate` simply sets these to be fixed constants, so we make slight modifications and assumptions for convenience of analysis.

Let $\varepsilon = (0, 1)$ be an error parameter, and let c be a sufficiently large constant.

• We set d_{thr} to be the largest d such that $\sum_{r \geq d} g_h(r) \geq (c(\log n)/\varepsilon^2)/p_h$. (In the implementation, we hardcoded c/ε^2 to be 50.)

• We assume that p_t is chosen so that $d_{thr} \geq \log(1/\varepsilon)/p_t$.

We begin with the analysis of the head estimator, which is a straightforward Chernoff bound application.

Lemma 5. *For all* $d \leq d_{thr}$, $\mathbf{E}[\widehat{N}(d)] = N(d)$. *With probability* $> 1 - 1/n$, *for all* $d \leq d_{thr}$, $|\widehat{N}(d) - N(d)| \leq \varepsilon N(d)$.

Proof. Fix some $d \leq d_{thr}$. Note that the head estimator is used for $\widehat{N}(d)$. Also, $\sum_{r \geq d} g_h(r)$ is precisely the number of vertices of degree at least d in S_h. For convenience, denote this by X_d, and observe that it is monotonically decreasing in d. By Lemma 1, each vertex is added independently to S_h with probability p_h. Thus, $\mathbf{E}[X_d] = p_h \cdot N(d)$. Note that $\widehat{N}(d)$ is precisely X_d/p_h, so $\mathbf{E}[\widehat{N}(d)] = N(d)$.

Since d_{thr} is itself a random variable, we need a little care to prove the lemma. Observe that X_d is well-defined for all d, and is the sum of Bernoulli random variables. By a multiplicative Chernoff bound (refer to Theorem 1.1 in [41]), $\Pr[|X_d - \mathbf{E}[X_d]| \leq \varepsilon \mathbf{E}[X_d]] \leq 2\exp(-\varepsilon^2 \mathbf{E}[X_d]/3)$. Furthermore, by an alternate bound, if $B \geq e\mathbf{E}[X]$, then $\Pr[X \geq B] < 2^{-B}$.

When $\mathbf{E}[X_d] = p_h \cdot N(d) \geq (c(\log n)/3\varepsilon^2)$, apply the first bound. When $\mathbf{E}[X_d] < (c(\log n)/3\varepsilon^2)$, apply the second bound with $B = c(\log n)/\varepsilon^2$. Finally, we apply the union bound over all errors, which a calculation shows to be $< 1/n$. Hence, for any d where $\mathbf{E}[X_d] < c(\log n)/3\varepsilon^2$, $X_d < c(\log n)/\varepsilon^2$. So, d_{thr} must be smaller than any such degree. Thus, for all $d \leq d_{thr}$, $\mathbf{E}[X_d] \geq c(\log n)/3\varepsilon^2$, and the first Chernoff bound gives the desired concentration. □

The more challenging part is to analyze the tail estimator. We fall short of giving a complete proof that it works. Nonetheless, we provide some mathematical evidence of its correctness. We provide a high level explanation of the math that follows. We warn the reader that we shall switch between estimates for $N(d)$ and $n(d)$.

The weakness of the head estimator is made clear in the proof of the previous lemma. The Chernoff bounds says that the error probability of estimating of $N(d)$ is roughly $\exp(-p_h \cdot N(d))$. This goes to 1 as $N(d)$ becomes smaller than $1/p_h$. That is precisely what happens in the tail of the degree distribution, which contains fewer vertices of higher degree. In general, mild fluctuations in

estimates for low degree vertices is ok (there are many of them), but even a little wagging in the tail estimates creates significant error.

But high degree vertices are more likely to be in S_t by Lemma 2. Let $S_t(d)$ denote the subset of degree d vertices in S. We show in Lemma 6 how to get an estimate of $n(d)$ from $|S_t(d)|$, where the error probabilities are roughly $\exp(-p_t \cdot d \cdot n(d))$. *Note the extra d factor.* As long as $d \cdot n(d) \geq 1/p_t$, we can hope for concentration. In other words, even though high degree vertices are infrequent, it is provably possible to get accurate estimates for these counts.

Unfortunately, it is not clear how to estimate $|S_t(d)|$, since $ct_t(v)$ is quite different from d_v. As mentioned earlier, we make the (admittedly erroneous) assumption that $ct_t(v) = d_v - \mathbf{E}_{X \sim TG_{p_t,d_v}}[X]$, based on Lemma 2 and Lemma 3. This is used to predict the actual degree of $v \in S_t$, based on $ct_t(v)$. While this assumption is wrong because the truncated geometric distribution has large variance, in practice, it works quite well.

In `estimate`, the proxy for $|S_t(d)|$ is given by $\widetilde{C}_t(d)$. We show that the "ccdh" (or partial sums) of $\widetilde{C}_t(d)$ approximates those of $|S_t(d)|$. In other words, we can a get a rough approximation for the number of vertices of degree at least d in S_t. This is what is proven in Theorem 7 and the subsequent calculations.

We now proceed with the formal proofs. The following lemma provides an appropriate concentration bound for estimating $n(d)$ from $|S_t(d)|$.

Lemma 6. *For all d, $\mathbf{E}[|S_t(d)|] = (1 - (1 - p_t)^d)n(d)$. For all $d \geq d_{thr}$ and sufficiently small p_t: with probability at least $1 - 2\exp(-\varepsilon p_t \cdot d \cdot n(d)/16)$, $\left||S_t(d)| - \mathbf{E}[|S_t(d)|]\right| \leq \varepsilon \mathbf{E}[|S_t(d)|]$.*

Proof. Every degree d vertex is added to S with probability $1 - (1 - p_t)^d$ (for convenience, denote this by α). Linearity of expectation proves that $\mathbf{E}[|S_t(d)|] = \alpha n(d)$. Note that $|S_t(d)|$ is the sum of Bernoulli random variables, each with expectation α. By the original Chernoff-Hoeffding bound [42], $\Pr[|S_t(d)| \leq (1 - \varepsilon)\alpha n(d)] \leq \exp(-D(\alpha(1 - \varepsilon)\|\alpha)n(d))$, where $D(\cdot, \cdot)$ denotes the KL-divergence. With some manipulations,

$$D(\alpha(1-\varepsilon)\|\alpha) = \alpha(1-\varepsilon)\ln\frac{\alpha(1-\varepsilon)}{\alpha} + (1 - \alpha(1-\varepsilon))\ln\frac{1 - \alpha(1-\varepsilon)}{1-\alpha}$$
$$\geq \alpha\ln(1-\varepsilon) + \alpha\varepsilon\ln(1 + \alpha\varepsilon/(1-\alpha))$$

Now we use $d \geq d_{thr} \geq \log(1/\varepsilon)/p_t$. A calculation yields $[1 - (1 - p_t)^d]\varepsilon/(1 - p_t)^d \geq 1/2$ for sufficiently small p_t. Hence, the expression above is bounded below by:

$$-2\varepsilon + \alpha\varepsilon\ln(\alpha\varepsilon/(1-\alpha))/4 \geq -2\varepsilon + \alpha\varepsilon\ln(\alpha\varepsilon)/4 - \alpha\varepsilon\ln(1-p_t)^d/4$$
$$\geq -4\varepsilon + \varepsilon p_t d/8 \geq \varepsilon p_t d/16$$

An analogous bound holds for the upper tail, and a union bound completes the proof. $\square$

Hence, we would like to estimate $|S_t(d)|$ and divide by $1 - (1 - p_t)^d$ to get estimates for $n(d)$ (where d is large). Our estimate for $|S_t(d)|$ is $\widetilde{C}(d)$, and this scaling is precisely what is done in `estimate`.

Definition 2. • $C_{p_t,s}$: *The cdf of $TG_{p_t,s}$, formally $C_{p_t,s}(k) = \Pr_{X \sim TG_{p_t,s}}[X \leq k] = [1 - (1 - p_t)^{k+1}]/[1 - (1 - p_t)^s]$.*
• $\ell(d) = \lfloor \mathbf{E}_{X \sim TG_{p_t,d}}[X] \rfloor$.

9

- $red(d) = d - \ell(d)$.

Indeed, we will show that the "ccdh" of $|S_t(d)|$ is somewhat approximated by that of $\widetilde{C}(d)$.

Theorem 7. $\mathbf{E}[\sum_{r \geq d} \widetilde{C}(d)] = \sum_{r \geq red(d)} C_{p_t,r}(r - red(d))\mathbf{E}[|S_t(r)|]$.

Proof. Note that $\ell(d)$ is monotonically increasing in d. Any $v \in S$ such that $ct_t(v) \geq red(d)$ will be counted as part of $\widetilde{C}(r)$, for some $r \geq d$. The quantity $loss(v) = d_v - ct_t(v)$, conditioned in $v \in S$, is distributed as TG_{p_t,d_v}. The probability of the loss being most $d_v - red(d)$ is exactly $C_{p_t,d_v}(d_v - red(d))$.

$$
\begin{aligned}
\mathbf{E}[\sum_{r \geq d} \widetilde{C}(d)] &= \sum_v \Pr[v \in S] \Pr[loss(v) \leq d_v - red(d) | v \in S] \\
&= \sum_v [1 - (1 - p_t)^{d_v}] C_{p_t,d_v}(d_v - red(d)) \\
&= \sum_{r \geq red(d)} C_{p_t,r}(r - red(d))[1 - (1 - p_t)^d] n(r)
\end{aligned}
$$

By Lemma 6, $[1 - (1 - p_t)^r]n(r) = \mathbf{E}[|S_t(r)|]$ $\qquad\qquad\square$

3.2 Making sense of Theorem 7

Fix p_t and d. Consider $C_{p_t,r}(r - red(d))$ as a function of r, and suppose it had value 0 for $r < d$, and value 1 for $r \geq d$. Think of this as the ideal value for this function. Then, by Theorem 7, $\mathbf{E}[\sum_{r \geq d} \widetilde{C}(d)] = \sum_{r \geq d} \mathbf{E}[|S_t(r)|]$, which would be exactly what we want. We prove that the "coefficients" $C_{p_t,r}(r - red(d))$ behave like a step function with a transition roughly at d. So $\mathbf{E}[\sum_{r \geq d} \widetilde{C}(d)]$ is a sort of smoothed version of $\sum_{r \geq d} \mathbf{E}[|S_t(r)|]$.

We begin with some approximations for $C_{p_t,r}$. It is useful to think of the limit as $p \to 0$ and reparametrize as $d = k/p_t$. By Lemma 3,

$$
\begin{aligned}
\mathbf{E}_{X \sim TG_{p_t,d}}[X] &= \frac{1 - p_t - (1 - p_t)^{1+k/p_t} - k(1 - p_t)^{k/p_t}}{p_t(1 - (1 - p_t)^{k/p_t})} \\
&\approx \frac{1 - p_t - e^{-k} - ke^{-k}}{p_t(1 - e^{-k})} \\
&\approx \frac{1}{p_t} - \frac{ke^{-k}}{p_t(1 - e^{-k}} \approx \frac{1}{p_t}(1 - ke^{-k})
\end{aligned}
$$

Thus, $red(d) = k/p_t - 1/p_t + ke^{-k}/p_t$. Now consider some $r = x/p_t \geq red(p)$.

$$
\begin{aligned}
C_{p_t,r}(r - red(d)) &= \frac{1 - (1 - p_t)^{1+(1/p)\cdot(x-k+1-ke^{-k})}}{1 - (1 - p_t)^{x/p}} \\
&\approx \frac{1 - \exp(-(x - k + 1 - ke^{-k}))}{1 - \exp(-x)} \quad\quad (1)
\end{aligned}
$$

Clearly, as x becomes large, this expression goes to 1. The minimum possible value of x is $k - 1 + ke^k$ (equivalently, $r = red(d)$), for which the expression is 0. It behaves roughly like a step
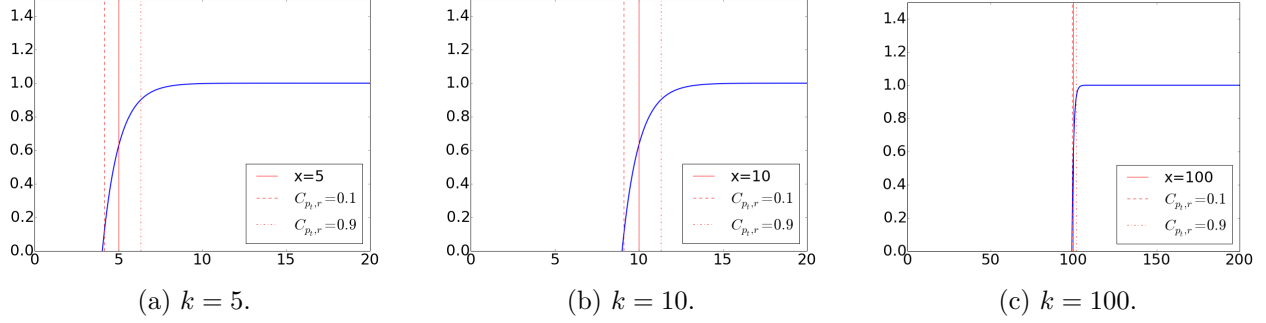
10

Figure 2: Plots of $C_{p_t,r}$ according to (1) for different values of k. Note that r is set to x/k. In each plot, the thin vertical line is $x = k$, and the dashed and dotted lines correspond to values of 0.01 and 0.09, respectively.

function, with a transition point (roughly) at $k-1+ke^{-k}$. As k becomes large, the transition point is $k-1$, close to k. When k is small, the extra ke^{-k} additive terms ensures the transition is closer to k. Of course, as k becomes smaller, the function looks less like a sharp transition function. This is shown in Fig. 2. We plot $C_{p_t,r}$ according to (1) for $k = 5, 10, 100$. The red vertical line is $x = k$ (so $r = k/p_t$), and we draw dashed vertical lines corresponding to value 0.1 and 0.9. The width between the dashed lines is a rough measure of the error in approximation. Observe how it is fairly close to a step function for $k = 10$, and is a coarser approximation for $k = 5$.

Hence, $\mathbf{E}[\sum_{r \geq d} \widetilde{C}(r)]$ is much further from $\mathbf{E}[\sum_{r \geq d} |S_t(r))|]$, and $\texttt{estimate}$ provides worse results. But we set $d_{thr} > \log(1/\varepsilon)/p_t$. So for degrees close to $1/p_t$, we do not use the tail estimator.

4 The Relative Hausdorff distance

One of the main challenges in experimentally validating the behavior of $\texttt{estimate}$ is in defining a distance between ccdhs. As we hinted earlier, existing statistical distances do not capture "similarity" of ccdhs. Motivated by concerns (detailed below), we define a new notion of distance between ccdhs (technically, between cumulative complementary histograms). This is inspired by the geometric notion of Hausdorff distance between subsets of a metric space. We say a ccdh is non-trivial if it contains some non-zero point.

Definition 3. *Let F and G be non-trivial ccdhs. Fix non-negative numbers ε, δ. The distributions F and G are (ε, δ)-close by Relative Hausdorff (RH) distance if:*

$$\forall d, \exists d' \in [(1-\varepsilon)d, (1+\varepsilon)d], \ such \ that \ |F(d) - G(d')| \leq \delta F(d).$$

(An analogous condition holds with F and G switched.)

The RH-*distance between F and G (denoted $RH(F, G)$) is $\inf\{\varepsilon | F$ and G are $(\varepsilon, \varepsilon)$-close$\}$.*

Note that the RH-distance can be greater than 1. For $\varepsilon' \geq \varepsilon$ and $\delta' > \delta$, if F and G are (ε, δ)-close, they are also (ε', δ')-close. Since F and G are non-trivial, we can set ε to be large enough so that for some δ, F and G are (ε, δ)-close. Thus, the RH distance always exists. If $RH(F, G) = 0$, then F and G are identical.

Observe that RH distance tolerates error both in degree and frequency, which is very important for comparing degree distributions. The RH distance exactly captures the notion of being close

11

in log-scale, but is a much more stringent condition. It forces *all* points in F to be close to some point in G (and vice versa). All "outlier" and tail behavior in F must be approximated in G. For RH-close ccdhs, the maximum degrees must be close, and furthermore, there must be approximate agreement for frequencies at all scales.

To understand numerics, we think it is useful think of an RH-distance < 0.05 to be quite small. Suppose $RH(N, \widehat{N}) < 0.05$ for a true ccdh N and our algorithm output $\widehat{N}$. This means that for every reported point $\widehat{N}(d)$ is within 5% of some $N(d')$, where d' is within 5% of d (and vice versa). Any RH distance greater than 1 is very large, since we only get closeness when $\varepsilon \geq 1$.

4.1 Problems with KS-statistic

Fix two ccdhs F and G. A standard comparison metric is the Kolmogorov-Smirnov (KS) statistic, $KS(F, G) = \max_x |\widetilde{F}(x) - \widetilde{G}(x)|$, where $\widetilde{F}, \widetilde{G}$ are normalized as distributions. (So $\widetilde{F}(d)$ is the fraction of vertices with degree at least d.)

We discuss specific problems with the KS statistic and show how RH avoids these pitfalls. (The exact same issues also holds for normed distances, so we do not explicitly calculate these.)

Comparing cliques: Let F be the ccdh of an n-clique and G be the ccdh of an $(n-1)$-clique. So $\forall\ 0 \leq i \leq n-1, F(i) = n, \forall\ 0 \leq i \leq n-2, G(i) = n-1$, and all other values are 0. The KS-statistic is actually 1 (which is extremely large), since $\widetilde{G}(n-2) = 0$ but $\widetilde{F}(n-2) = 1$. This is inconsistent with our intuitive notion that these degree distributions are similar. The RH distance is $O(1/n)$, since it allows for error in degree and frequency.

Star vs matching: Let F be the ccdh of a star with n vertices, and G be the ccdh of a matching (disjoint edges) with n vertices. (Assume n is even.) So $F(1) = n, \forall 2 \leq i \leq n-1, F(i) = 1$, and other values are 0. We also have $G(1) = n$, and all other values are 0. The values of F that are 1 are insignificant compared to the dominant $F(1) = n$. A calculation shows $KS(F, G) = O(1/n)$, though we should probably consider them different. On the other hand, $RH(F, G) = 1 - \Theta(1/n)$. The "outlier" $F(n-1) = 1$ forces the ε to be $1 - \Theta(1/n)$, since $G(i) = 0$ for $i > 1$.

Ignoring the tail: Let F be the ccdh of the as-Skitter graph, as plotted in Fig. 1a. Let G be the same ccdh up to degree 100 and zero afterwards. In other words, G is identical to F up to the "tail" starting at degree 100. The fraction of vertices with degree > 100 is at most 0.01. A calculation shows that $KS(F, G) < 0.01$. So ignoring a large portion of the tail still yields small KS-distance. The RH-distance is > 0.99, since ε needs to be large to handle the tail of F.

5 Experimental Results

We implemented the algorithm in Python and performed experiments on a Samsung NP-QX411L laptop with an Intel Core i5-2450M 2.5GHz four core processor and 5.7GB of memory. To simulate a stream, we convert a graph to a list of edges stored in a text file, and read the file one line at a time. In the case that the graph is directed, we treat it as undirected by considering each edge as an unordered pair of vertices. Note that this may imply multi- or parallel edges, though we calculate degrees for the actual ccdh respecting this notion.

We test the algorithm on a number of graphs from the SNAP [16] and KONECT [43] collections, the statistics of which are summarized in Table 1. We use the as-Skitter graph on 1.7M nodes and 11M edges as a case study.
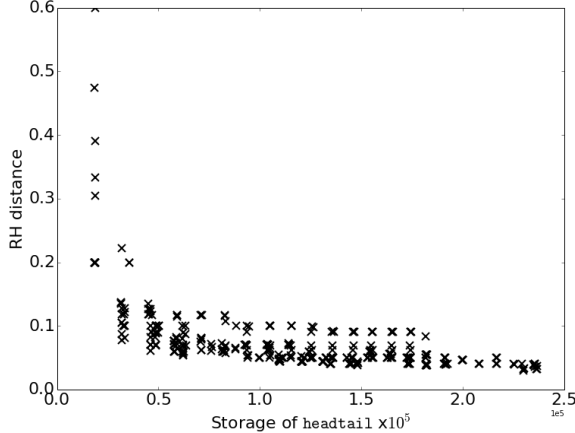
Figure 3: RH distance as the storage of `headtail` increase.

We use the phrase *storage of* `headtail` to indicate the total storage $|S_h| + |S_t|$. As explained in Theorem 4, this depends on p_h and p_t.

5.1 Convergence of `headtail`

We demonstrate how increasing the storage of `headtail` leads to convergence of the ccdh. We fix the as-Skitter graph. We increase the storage by letting p_h range from 0.01 to 0.1 in increments of 0.01, and p_t range from 0.01 to 0.16 in increments of 0.01. For each setting of p_h and p_t, we perform five independent runs of `headtail`. We also run ten independent runs fixing $p_h = 0.005, p_t = 0.01$. For each such run, we compute the RH distance between the output of `headtail` with the true ccdh. The results are shown in Fig. 3. Observe how the RH distance goes to zero as the storage increases. In particular, `headtail` outputs a ccdh with RH distance as small as 0.03 using 230K space.

We do a more nuanced study of how p_h and p_t affect convergence. In this experiment, we fix a value p_h and vary p_t in increments of 0.02. We repeat this process for $p_h = 0.01, 0.025, 0.05, 0.075, 0.1$. The RH distances of the runs are plotted in Figure 4. Each line in the plot corresponds to a fixed p_h value, and the RH distances are plotted against p_t. We point out that an RH distance of about 0.04 is achieved with head and tail probabilities as small as 0.025, 0.03, respectively, resulting in a total sample size of 82K or 0.7% of the edge stream. For each fixed p_h, increasing p_t initially decreases the RH distance, but it eventually converges to a non-zero value. This is because all the error is coming from the head estimate. As we increase p_h, the convergence value goes down to zero, as expected.

5.2 Results for various graphs

Here we demonstrate the quality of the estimates output by `headtail` on a variety of graphs. Each of the graphs are from the SNAP graph collection [16] with the exception of the youtube and youtube-friendship graphs which are from the KONECT [43] collection. The node and edge set sizes of each graph are given in the second and third columns of Table 1, respectively. For each graph we include the storage of the algorithm and the RH distance of the estimate for two example runs. The storage is less than $< 1\%$ in almost at runs, and certainly less than $< 2\%$. Observe how
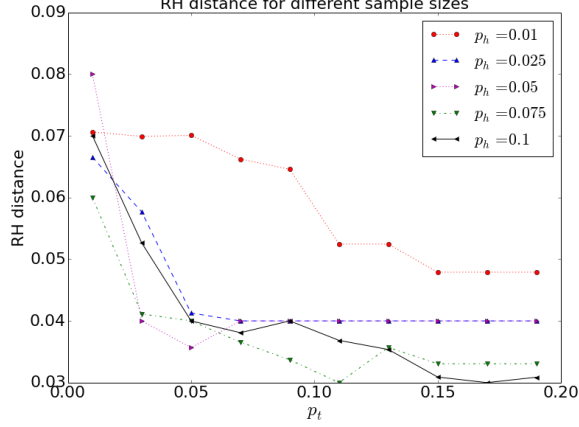
Figure 4: RH distance of the estimate output by our algorithm as p_h and p_t vary. Each line in the plot correponds to a fixed value for p_h, and plots the RH distance as p_t varies. A near optimal RH value is achived with $p_h = 0.025$ and $p_t = 0.03$, which yielded sample sets with $|S_h| + |S_t| \approx 0.007m$.

the RH distance is usually less than 0.1. In our worst examples, (soc-Pokec and com-Orkut), the RH distance is less than 0.15. We stress that RH distance is a rather stringent condition, since it requires closeness of the estimate at *all* degrees.

In Fig. 1 of the introduction, we have plotted the actually ccdh and the output of `headtail` for three of these graphs. Observe the near identical match in all examples.

5.3 Errors at different scales

Here we investigate how well `headtail` performs at different scales. Specifically, we measure the error of a ccdh estimate at each degree. Let N be the ccdh of the as-Skitter graph, and $\widehat{N}$ be the `headtail` output. The RH distance is maximized over all degrees, so we do a more detailed analysis of the estimate errors. We fix a value for ε and for each degree d compute the minimum value δ such that $\exists d' \in [(1 - \varepsilon)d, (1 + \varepsilon)d]$ where $|N(d) - \widehat{N}(d')| \leq \delta N(d)$ and vice versa. In words, we are "opening up" the definition of RH-distance and looking at the profile for every degree.

We performed a run of `headtail` with $p_h = 0.01$ and $p_t = 0.0007$ for the as-Skitter graph. This used a storage of 31K ($< 0.5\%$ of stream). We then plot in Fig. 5 the corresponding δ values with ε set to 0.1 . The red 'x' markers denote the δ-values for `headtail` (the other markers are explained later). Observe how the δ values are quite small throughout, and peak at degree 100 to roughly 0.08. In this case, `headtail` achieves an RH-distance of about 0.1 with 31K space.

5.4 Comparing to other methods

While there is no existing small-space algorithm that has demonstrable convergence to the ccdh, there are numerous algorithms to only capture the tail. These are classic "heavy hitters" algorithms: the *frequent* algorithm [20, 21, 22], the *lossy counting* algorithm [23], and the *space saving* algorithm [24]. We study the performance of these methods. For convenience, we use "head estimator" to denote the algorithm that simply takes uniform samples of vertices and uses their degrees to estimate the full ccdh. This is basically what `headtail` employs for $d \leq d_{thr}$.

14

Graph	n	m	Space	RH distance
youtube	1.1M	3M	21K	0.1
			90K	0.076
wiki-Talk	2.3M	5M	38K	0.1
			74K	0.055
youtube-friendship	3M	9M	80K	0.067
			196K	0.05
as-Skitter	1.7M	11M	31K	0.1
			69K	0.073
soc-Pokec	1.6M	30M	75K	0.29
			212K	0.14
com-LiveJournal	4M	34M	335K	0.08
			467K	0.058
com-Orkut	3M	117M	273K	0.14
			387K	0.13

Table 1: Performance of `headtail` for a number of graphs.

Figure 5: RH distances at different degrees. We plot the δ-distance for $\varepsilon = 0.1$. The red 'x' markers correspond to an estimate output by `headtail` using a storage of 31K. The estimate is $(0.1, 0.08)$-far from the true ccdh. The rest of the plots correspond to combinations of the head estimator using 17K space and the heavy hitter algorithms using 34K space for a total of 51K space. The *lossy counting* estimate is $(0.1, 1.5)$-far from the true ccdh, the *space saving* estimate $(0.1, 0.4)$-far and the *frequent* estimate is $(0.1, 0.33)$-far from the true ccdh.

We fix the as-Skitter graph, and set the storage used by these algorithms to 35K. (Note that with storage 31K, `headtail` gives an estimate with RH-distance less than 0.1.) We show the resulting estimates of these algorithms in Fig. 6. Not surprisingly, none of these algorithms give reasonable estimates for $N(d)$, where $d \leq 10^3$.

At the face of it, the above algorithms perform reasonably well on the tail. The head estimator (which is quite simple) seems to work well for the head. Could we just combine these algorithms, and outperform `headtail`? We show that this is not the case. Crucially, none of these algorithms actually get accurate estimates even at the moderate to high degrees, despite the apparent closeness in the log-log plot of Fig. 6.

We convert the existing algorithms for the full ccdh, by combining with the head estimator. Pick (say) the algorithm *frequent*. We first run the head estimator with 20K space. We choose an appropriate d_{thr}, where we apply the head estimator for $d \leq d_{thr}$, and *frequent* for $d > d_{thr}$. We pick the d_{thr} that minimizes the RH distance to $\{N(d)\}$. We do the same for each of *frequent*, *space saving*, and *lossy counting*. Note that we are being extra generous to the competing methods. First, the total storage used is about 50K. Furthermore, we choose the d_{thr} to minimize RH distance, while `headtail` chooses it based on a fixed formula.

The RH distance we achieved was 0.3 (*frequent*), 0.5 (*space saving*), and 1.5 (*lossy counting*). All of these used storage 50K. In contrast, `headtail` had RH distance of 0.1 with 31K storage. We measure the errors at all scales in Fig. 5, for all these algorithms. This is exactly using the explanation in previous section, by setting $\varepsilon = 0.1$, and plotting the δ values for all the estimates.

We immediately see how the δ-values (errors) for all the competing procedures are much higher than `headtail`. Indeed, for degrees around 10^3, the errors of the other procedures are extremely high, despite higher storage. We see that `headtail` handily beats all the procedures, at pretty much all scales simultaneously. In Fig. 7, we plot the output ccdh for the head estimator combined with *frequent*. As expected from Fig. 5, we see a fair amount of fluctuation from the true ccdh in the the intermediate to high degrees. We stress that a small fluctuation in a log-log plot is actually a fairly large error in the RH measure.

For completeness, we increase the storage of the competing methods to get RH distance of around 0.1. For all the other algorithms, we require storage more than 150K to get comparable error to what `headtail` gives with 31K storage.

5.5 Results for different stream orderings

As stated previously, our algorithms do not assume any stream order. In this section we test the performance of the algorithm when provided the stream in different orderings. We use six different orderings in total. The first three are different random orderings. The second three are each edgelists (that is, all the edges adjacent to a particular node are read in sequence), but the orderings of the nodes are different. In one, we read the nodes of highest degree first, in another we read the nodes in increasing order of degree, and in the last we consider a random ordering of the nodes. In each experiment we let $p_h = 0.01$ and $p_t = 0.04$. The standard deviation of the RH distances for each ordering is 0.009. Table 2 summarizes the RH distance of estimated ccdhs with different stream orderings.
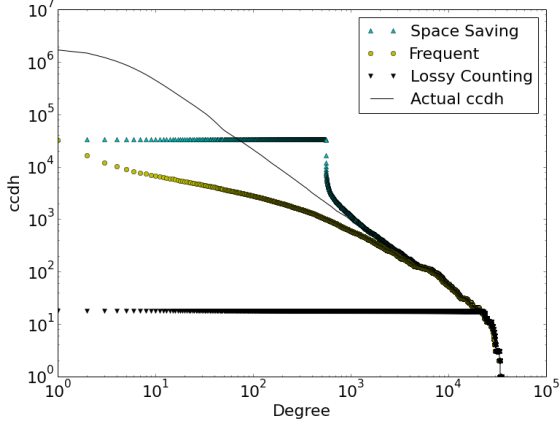
Figure 6: ccdh estimates output by the *frequent*, *lossy counting*, and *space saving* algorithms each using a storage of 35K.
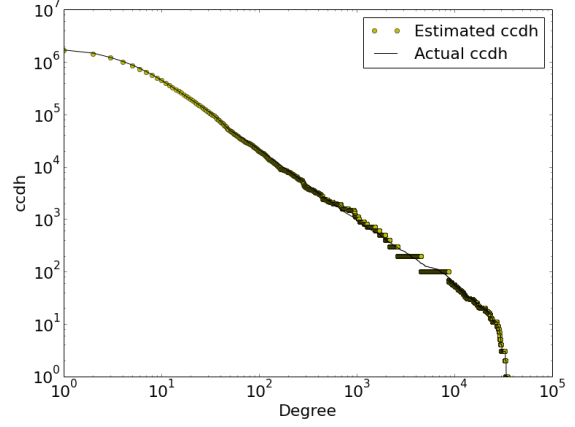

Figure 7: ccdh estimate output by the head estimator combined with the *frequent* algorithm using a storage of 50K. The RH distance is 0.33.

Ordering	RH distance
Random1	0.068
Random2	0.06
Random3	0.07
Edgelist: Decreasing order of degree	0.08
Edgelist: Increasing order of degree	0.083
Edgelist: Random	0.061

Table 2: Performance of `headtail` for different stream orderings. The first three are different random stream orderings. The second three are edgelists permuted by the nodes. In each trial $p_h = 0.01, p_t = 0.04$.

Acknowledgment

References

[1] H. Kwak, C. Lee, H. Park, and S. Moon, "What is twitter, a social network or a news media?" in *WWW*, 2010. 1

[2] http://facebook.com/press/info.php?statistics, 2012. 1

[3] M. Meiss, F. Menczer, S. Fortunato, A. Flammini, and A. Vespignani, "Ranking web sites with real user traffic," in *WSDM*. ACM, 2008, pp. 65–75. 1

[4] A.-L. Barabási and R. Albert, "Emergence of scaling in random networks," *Science*, vol. 286, pp. 509–512, Oct. 1999. 2

[5] M. Faloutsos, P. Faloutsos, and C. Faloutsos, "On power-law relationships of the internet topology," in *SIGCOMM*, 1999, pp. 251–262. 2

[6] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener, "Graph structure in the web," *Computer Networks*, vol. 33, pp. 309–320, 2000. 2

[7] R. Cohen, K. Erez, D. ben Avraham, and S. Havlin, "Resilience of the internet to random breakdowns," *Phys. Rev. Lett.*, vol. 85, no. 46268, 2000. 2

[8] M. E. J. Newman, S. Strogatz, and D. Watts, "Random graphs with arbitrary degree distributions and their applications," *Physical Review E*, vol. 64, p. 026118, 2001. 2

[9] D. Pennock, G. Flake, S. Lawrence, E. Glover, and C. L. Giles, "Winners don't take all: Characterizing the competition for links on the web," *Proceedings of the National Academy of Sciences*, vol. 99, no. 8, pp. 5207–5211, 2002. 2

[10] M. E. J. Newman, "The structure and function of complex networks," *SIAM Review*, vol. 45, no. 2, pp. 167–256, 2003. 2

[11] M. Mitzenmacher, "A brief history of generative models for power law and lognormal distributions," *Internet Mathematics*, vol. 1, no. 2, pp. 226–251, 2003. 2

[12] D. Chakrabarti and C. Faloutsos, "Graph mining: Laws, generators, and algorithms," *ACM Computing Surveys*, vol. 38, no. 1, 2006. 2

[13] C. Seshadhri, T. G. Kolda, and A. Pinar, "Community structure and scale-free collections of Erdös-Rényi graphs," *Physical Review E*, vol. 85, no. 5, p. 056109, May 2012. 2

[14] http://en.wikipedia.org/wiki/Scale-free_network. 2

[15] A. Clauset, C. R. Shalizi, and M. E. J. Newman, "Power-law distributions in empirical data," *SIAM Review*, vol. 51, no. 4, pp. 661–703, 2009. 2, 3, 4

[16] J. Leskovec, "Snap stanford network analysis project," http://snap.standord.edu, 2015. 3, 12, 13

[17] P. Indyk and D. P. Woodruff, "Optimal approximations of the frequency moments of data streams," in *STOC*, 2005, pp. 202–208. 2, 4

[18] D. M. Kane, J. Nelson, and D. P. Woodruff, "An optimal algorithm for the distinct elements problem," in *PODS*, 2010, pp. 41–5. 2, 4

[19] G. Cormode and M. Hadjieleftheriou, "Finding frequent items in data streams," *VLDB*, vol. 1, no. 2, pp. 1530–1541, 2008. 2, 4

[20] E. D. Demaine, A. López-Ortiz, and J. I. Munro, "Frequency estimation of internet packet streams with limited space," in *Proc. of ESA 2002*. Springer, 2002, pp. 348–360. 4, 14

[21] R. M. Karp, S. Shenker, and C. H. Papadimitriou, "A simple algorithm for finding frequent elements in streams and bags," *TODS*, vol. 28, no. 1, pp. 51–55, 2003. 4, 14

[22] R. Berinde, P. Indyk, G. Cormode, and M. J. Strauss, "Space-optimal heavy hitters with strong error bounds," *ACM Trans. Database Syst.*, vol. 35, no. 4, p. 26, 2010. 4, 14

[23] G. S. Manku and R. Motwani, "Approximate frequency counts over data streams," in *VLDB*. VLDB Endowment, 2002, pp. 346–357. 4, 14

[24] A. Metwally, D. Agrawal, and A. E. Abbadi, "Efficient computation of frequent and top-k elements in data streams," in *Database Theory-ICDT 2005*. Springer, 2005, pp. 398–412. 4, 14

[25] M. Charikar, K. Chen, and M. Farach-Colton, "Finding frequent items in data streams," in *Automata, Languages and Programming*. Springer, 2002, pp. 693–703. 4

[26] G. Cormode and S. Muthukrishnan, "An improved data stream summary: the count-min sketch and its applications," *Journal of Algorithms*, vol. 55, no. 1, pp. 58–75, 2005. 4

[27] N. Alon, Y. Matias, and M. Szegedy, "The space complexity of approximating the frequency moments," *J. Comput. Syst. Sci.*, vol. 58, no. 1, pp. 137–147, 1999. 4, 5

[28] F. Korn, S. Muthukrishnan, and Y. Wu, "Modeling skew in data streams," in *SIGMOD*. ACM, 2006, pp. 181–192. 4

[29] A. McGregor, "Graph stream algorithms: A survey," *SIGMOD*, vol. 43, no. 1, pp. 9–20, 2014. 4

[30] M. Jha, C. Seshadhri, and A. Pinar, "A space efficient streaming algorithm for triangle counting using the birthday paradox," in *SIGKDD*. ACM, 2013, pp. 589–597. 4

[31] A. Pavan, K. Tangwongsan, S. Tirthapura, and K. Wu, "Counting and sampling triangles from a graph stream," *PVLDB*, vol. 6, no. 14, pp. 1870–1881, 2013. 4

[32] N. K. Ahmed, N. Duffield, J. Neville, and R. Kompella, "Graph sample and hold: A framework for big-graph analytics," in *SIGKDD*, ACM. ACM, 2014, pp. 1446–1455. 4, 5

[33] S. Guha, A. McGregor, and D. Tench, "Vertex and hyperedge connectivity in dynamic graph streams," in *PODS*, 2015, pp. 241–247. 4

[34] M. Kapralov, S. Khanna, and M. Sudan, "Approximating matching size from random streams," in *SODA*, 2014, pp. 734–751. 4

[35] A. McGregor, "Finding graph matchings in data streams," in *APPROX and RANDOM*, 2005, pp. 170–181. 4

[36] G. Cormode and S. Muthukrishnan, "Space efficient mining of multigraph streams," in *SIGACT-SIGMOD-SIGART*, 2005, pp. 271–282. 4

[37] N. Ahmed, J. Neville, and R. Kompella, "Reconsidering the foundations of network sampling," in *WIN 10*, 2010. 4, 5

[38] ——, "Space-efficient sampling from social activity streams," in *SIGKDD BigMine*, 2012, pp. 1–8. 4

[39] N. K. Ahmed, J. Neville, and R. Kompella, "Network sampling: From static to streaming graphs," *TKDD*, vol. 8, no. 2, p. 7, 2014. 4

[40] V. Braverman and R. Ostrovsky, "Approximating large frequency moments with pick-and-drop sampling," in *APPROX*. Springer, 2013, pp. 42–57. 5

[41] D. Dubhashi and A. Panconesi, *Concentration of Measure for the Analysis of Randomised Algorithms*. Cambridge University Press, 2012. 8

[42] H. Chernoff, "A measure of asymptotic efficiency for tests of a hypothesis based on the sum of observations," *Annals of Mathematical Statistics*, vol. 23, no. 4, pp. 493–507, 1952. 9

[43] J. Kunegis, "The koblenz network collection," http://konect.uni-koblenz.de, 2015. 12, 13